Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 16 (2014) 824 – 833

Procedia Technology

CENTERIS 2014 - Conference on ENTERprise Information Systems / ProjMAN 2014 - International Conference on Project MANagement / HCIST 2014 - International Conference on Health and Social Care Information Systems and Technologies

Process invariants: an approach to model expected exceptions

Pedro Ferreira[a,c,*], Ricardo Martinho[a,c], Dulce Domingos[b,c]

[a] *Departamento de Engenharia Informática, Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, 2411-901 Leiria, Portugal*
[b] *Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal*
[c] *Laboratório de Sistemas Informáticos de Grande-Escala, 1749-016 Lisboa, Portugal*

Abstract

Business processes benefit from context information provided by Internet of Things (IoT) technologies. However, business processes tend to have more and more conditions. Thus, the process modeler needs to include many expected exceptions in the process model. Consequently, it increases the number of variables and workflow complexity. The definition of such complexity is cumbersome, time consuming and deviates the process modeler from the main flow. In this paper we propose an alternative to model expected exceptions, allowing the process modeler to focus on the right process. We propose the process invariants concept, which allows a process modeler to define a set of conditions that must remain valid within a process scope. Additionally, we implement our proposal by extending BPMN with process invariants.

Peer-review under responsibility of the Organizing Committee of CENTERIS 2014.

Keywords: IoT business processes; expected exceptions; process invariants; BPMN.

1. Introduction

The IoT offers significant improvement to enterprise applications regarding business process management [19, 6]. Embedding environment gathered data in business processes allows decision making at high management level. For instance, logistics plays an important role in the supply chain business processes, dealing with the control and planning of all the factors that will have an impact on transporting [11]. The basic logistics functions are to transport

* Corresponding author. Tel.: +351-244-830-010; fax: +351-244-813-013.
E-mail address: pedro.ferreira@ipleiria.pt

Peer-review under responsibility of the Organizing Committee of CENTERIS 2014.
doi:10.1016/j.protcy.2014.10.032

the right goods and the right quantity and right quality at the right time to the right place for the right price [4]. Environment data (such as the temperature in a fruit container) obtained through, e.g., Wireless Sensor Networks (WSNs) allows the overall logistics processes to be observed with detail. Thus, real-time responsiveness to events and other actions can be achieved in a timely fashion. For instance, objects not reaching their destination in time due to routing may have influence on the quality and, therefore, their price.

Environment gathered data has the potential to enhance enterprises business processes. WSNs and other Internet of Things (IoT)-related technologies also face many challenges from a process perspective, namely in the modelling of highly complex processes. Typical solutions address this challenge by focusing their approach on the modelling of expected exceptions and deviations. However, this is cumbersome and drives away the modeler to convey dozens of exception combinations (and still failing to predict them all) within a certain process model. Therefore, our approach is to focus on the modelling of "the right process" instead: defining the conditions that guarantee the "good case scenario" along the process execution.

In this paper we present language constructs that allow a process modeler to define "the right process", using a set of conditions that must be "right" during the whole process execution. For instance, the process modeler wants to define that the temperature cannot rise above 15◦C during a transportation scenario. We name these conditions as process invariants. An invariant is a mathematical expression that represents one (or more) restrictions and remains constant through time [3].

In addition, we implement our proposal with the Business Process Model and Notation (BPMN). BPMN is a graphical standard language to model business processes [7]. However, its specification does not provide language constructs to define the process invariants we proposed. Therefore, we also present an extension to BPMN.

The reminder of this paper is organized as follows: section 2 describes a motivation example, section 3 proposes the concept of process invariant as an alternative to model expected exceptions, section 4 proposes an extension to BPMN regarding process invariants, section 5 discusses related work and section 6 concludes this paper.

2. Motivation Scenario

In this section we provide an example scenario that will guide us along the paper. Consider a transportation process where the process modeler wants to assure the quality of the goods within a truck container. During transportation, measures from several sensors are used to obtain the values of temperature, humidity and pressure. The quality of the goods depends on those environment conditions: temperature must maintain a value between 10 and 15◦C. While these conditions are true, the process can proceed according to its main logic and no additional actions need to be specified. These conditions define the "right process". If they break anywhere along the process scope, a warning is raised.

In a standard process modelling language such as BPMN, the process modeler needs to specify these conditions with additional workflow behavior. Figure 1 displays an example of what could be a BPMN process model for this scenario. Every 60 seconds (or other programmable event), a request message to read environment conditions is triggered. Then, temperature variables are assigned the values of captured environment data. By default, the process will terminate. However, if these variables break the established conditions, a suitable response must be executed. Namely, (1) the status of the goods is evaluated to decide whether or not the price will be updated; (2) the status of other containers is evaluated to decide whether or not the container can be changed; (3) the estimation of delivery time to nearby clients is calculated to decide whether or not the route can be changed.

The problem with this example is that conditions must be specified within the process workflow with additional flow elements. Instead of concentrating on the process logic, the process modeler focus is deviated to additional verifications which should be modeled as conditions to the process. Even if the occurrence probability is very low, they need to be modelled within every flow element, which makes it complicated and troublesome, since condition breakage can happen anywhere in the process. Additionally, even modelling all the expected deviations there is still a chance that an unpredicted exception occurs. For instance, the process modeler has to decide the best solution if conditions are breached, considering the client's expectation and the cost impact: change destination, alter route, switch container, remove perished goods or reduce price.

Using BPMN, the correct representation of these conditions along the process scope demands a set of gateways and condition expressions with a lot of alternative workflows. Therefore, our approach is to define these conditions once and at a higher (process) level.

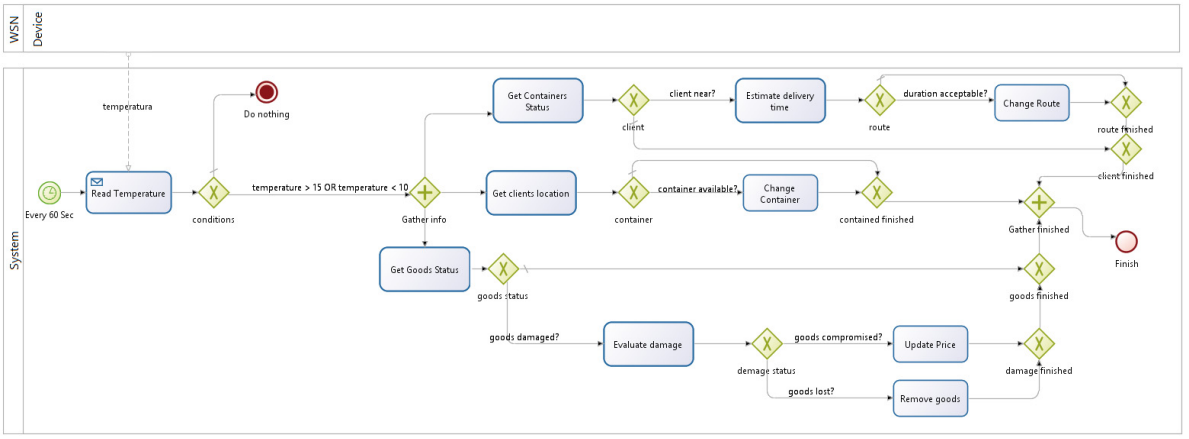

Fig. 1. Transportation scenario represented in BPMN.

3. Invariants

This section describes our approach regarding invariants in a business process. First, we propose the invariant concept as an alternative to model expected exceptions. Then we discuss the flexibility of BPMN regarding this concept and the related approaches that deal with process modelling for IoT. Finally, we discuss the invariants concept in other process languages.

3.1. Expected exceptions

Expected exceptions refer to predictable deviations from normal behavior of the process. These deviations can be addressed directly by adding alternative flow paths [20]. However, modelling exceptions is cumbersome. The process modeler focuses on alternative workflows and additional verifications instead of concentrating on the process logic. We propose an alternative to model expected exceptions as conditions to the process. We address and implement this idea with the invariants concept.

In the programming languages context, an assertion allows the definition of what it is assumed to always be true about data at a particular point in time. An assert statement evaluates a Boolean condition and does nothing if it is true. Additionally, it cannot go out of date. If it does, then it will fail in a legitimate case and the programmer is forced to update it. However, it is specific to a particular point in time. On the other hand, an invariant is a mathematical expression that represents one (or more) restrictions and remains constant through time [3].

Invariants are also known in Object Constraint Language (OCL). According to its specification, the OCL expression can be part of an Invariant, which is a constraint stereotyped as “invariant”. Additionally, all invariants in OCL are Boolean [8]. For instance, consider an object to model the container example in Figure 1. Such object can be modelled in a Unified Modeling Language (UML) class with the following property: Temperature (int). According to the OCL specification, the definition of an invariant consists in an expression that starts with the word context, followed by the class name and the token inv. The name of the invariant is optional. For instance, to represent the invariant Temperature > 15 or Temperature < 10 an appropriate OCL expression is written with the following syntax.

context c:Container **inv** myInvariant:

c.Temperature > 15 **or** c.Temperature < 10

A right business process is the one that considers a set of conditions valid within the overall process execution. These conditions assure the good case scenario as long as they are met. We name these conditions "process invariants". For instance, the condition "temperature remains higher than 15" can be represented with Temperature > 15 invariant.

3.2. BPMN flexibility

Flexibility is a known concern to business process modelling. In highly dynamic scenarios this concern is equally higher. There are different approaches that address flexibility. Most focus on the ability to adapt and the capability to change. However, some approaches focus on what needs to remain unchanged. Defining business process flexibility with the help of invariants provides the unchanging aspects of an enterprise as the need to maintain a constant identity [9]. This approach strengthens our research, as we embrace the conditions that need to be maintained constant within a process. However, there is no relation with any modelling language. Therefore, there is no concretization of the idea.

Other approaches address business process flexibility with business rules. Ensuring compliance in business process is a hard task and raises several challenges [15]. Thus, some approaches define business rules that guarantee the process is accordingly [14, 16, 17]. However, these focus on identifying patterns and defining them as part of the process. Therefore, the complexity of the process workflow is increased. On the other hand, a method for validating the compliance of business processes with business rules demonstrates this approach with Unified Modelling Language (UML) and OCL [13]. Araujo et al. use UML annotations to define the business rules and associate them with a specific activity within the process model. However, the business rules are not valid throughout the whole process.

Currently, BPMN is the best modelling solution regarding IoT technology, properties and resources [6]. However, it lacks language constructs to define process invariants, required by IoT-aware business processes as, for instance, business processes that implement logistics functions. BPMN does not foresee the "invariant" concept within its language constructs, using business rules as their closest concept. A BPMN business rule is a task that provides a mechanism for a process to interact with a Business Rules Engine (BRE). It allows the process to send data and receive data from the BRE to get the output from calculations. After a business rule task, the process modeler adds an exclusive gateway that determines the flow of the process based on the value of a data object that contains the result of running the business rule task [7].

Our goal is to add invariants to a BPMN process. However, there is a set of issues with this approach. First, the Business Rule task is a flow object. Hence, it is indicated to model process workflow and it does not allow the process modeler to add an invariant that monitors the workflow itself. Additionally, the business rule is a task and can only be used in a specific point of the process. This does not allow the process modeler to add an invariant to the overall process scope. Moreover, the task alone is not enough. It requires another flow object (gateway) to evaluate the output.

Second, the implementation and execution of the business rule depend on an engine. Thus, the process modeler is unable to specify the invariant to the process, as it depends on the rules engine and its own language. Finally, using a business rule task with a gateway to implement a condition to the process leads to the typical approaches issue: focusing the modelling on exception handling instead of the "right" process workflow itself. Thus, modelling exceptions is hard and time consuming, drifting process modelers from their main modelling goal.

Finally, the business rule task allows the process modeler to reduce the graphical process complexity by "hiding" some of the process workflow. From a graphical notation point on view, the process model does not reflect which conditions are actually being evaluated. To show such conditions within a process model, the process model must use an annotation to display the rules content.

3.3. IoT process modelling

Recently, the IoT attracted much attention from enterprises in different areas. Especially, the potential of IoT technologies concerning the representation of real work objects as participating entities in business process. However, integrating IoT technology into business processes requires the representation of IoT specific properties in

a graphical process notation. BPMN provides coverage for more IoT specific properties than any other existing business process modelling standard [6]. This analysis strengthens our approach in two aspects: (1) business process modelling standards require specific IoT concepts to represent IoT business processes, (2) BPMN is the most suitable to represent them. However, Meyer et al. do not specify a way to implement each of the properties outlined in their analysis. On further approaches, the authors integrate IoT devices as business process resources by adding IoT devices as a process resource type to BPMN [11, 12]. Indeed, these add IoT expressiveness to the modelling language. However, in order to easily monitor invariants these constructs face the same issues as BPMN business rule task. An exclusive gateway to evaluate the result of a condition is necessary. Instead of a specific task and gateway within the process our approach focuses on invariants to the process, allowing the process modeler to explicitly define conditions that guarantee the process right way.

Other approaches like [2] focus on IoT resources as well. However, it does not add expressiveness to the modelling language to represent IoT resources at process level. Instead, it focuses on representing IoT processes with BPMN primitives and compiling them into efficient event-based code that runs on the resources. Therefore, the process modeler cannot explicitly add IoT conditions to the process.

3.4. Invariants in other languages

Web Services Business Process Execution Language (WS-BPEL) is the standard reference for modelling executable business processes. However, the language specification is represented in a complex XML document and is hard to model business processes with. Therefore, also hard to model IoT business processes, even for process modelers. Still, there are several approaches suggest adding OCL to validate WS-BPEL specification [10]. OCL provides the constructs to add natural language constraints to WS-BPEL standard. However, the authors do not use OCL to add process constraints. Neither do they use OCL to define constraints that could represent IoT specific conditions required to model logistic functions.

Service oriented applications are highly dynamic and easily change at run-time. Current solutions overload the process model with alternative workflows that increase its complexity and defocus the process modeler from the real process. This issue is addressed with external monitoring rules to dynamically control the execution of the business process [1]. This approach relies on two important aspects, location and expressions. Both concepts are significant when dealing with invariants. Regarding location, an invariant monitoring rule indicates the WS-BPEL scope. Regarding expressions, an invariant expresses conditions on variables visible within the indicated WS-BPEL scope. These two aspects strengthen our research as we agree that process invariants must be defined within a scope. However, this approach implies the existence of an external engine rule, as the BPMN business rule task does. Thus, stating that this approach allows the process modeler to simply focus on the right process is arguable, at least.

4. Invariants BPMN extension definition

This section presents the BPMN extension. First, we introduce the invariant construct which allows the definition of expressions within a process scope. As the expression needs to have access to context information, we also extend BPMN with the Context Object construct to have updated information about context. Finally, we present the graphical notation to represent a process invariant.

4.1. Invariant Construct

To add invariants to the BPMN 2.0 notation, we extend the FlowElement class with the new “Invariant” subclass. Additionally, we associate the subclass with the Expression class and map it with the condition expression role. Figure 3 illustrates the UML class diagram representation of the proposed extension. FlowElementsContainer, Process, FlowElement and Expression represent classes that already exist within the BPMN 2.0 specification. In Figure 3, they are illustrated with a white background. The new “Invariant” class it is illustrated with a grey background. Our goal is to define invariants with BPMN at the process level. Therefore, we considered more than one approach to achieve this. (1) Add invariants to a LaneSet. A lane is a partition element that references multiple flow elements (FlowNodes) such as Activities, Events or Gateways. However, the process can be associated with

more than one LaneSet. Therefore, there is no guarantee that the invariant is known within the overall process scope. (2) Add invariants directly to the process. A process in BPMN is defined as a subclass of a flow element container for one or more sets of lanes. However, this approach requires an extension of the process itself. Thus, the invariants would also become a container of flow elements. (3) Add invariants to the process with a FlowElement extension. A FlowElement is an abstract super class for all elements that can appear in a Process flow. Unlike the FlowNode, the FlowElement does not have an incoming and outgoing reference. This approach is promising and introduces a lot of freedom, allowing the definition of invariants everywhere in the process, with the desired scope. Therefore, we chose the FlowElement class to contain the declaration of invariants to the process.

Unlike OCL, BPMN does not support the concept of invariant expression. However, other BPMN base elements can be used to represent them. The Expression element is used to specify an expression using natural-language text. These Expressions are not executable and are considered underspecified. The formal expression class is used to specify an executable Expression using a specified Expression language. A natural-language description of Expression can also be specified, in addition to the formal specification. These expressions are commonly used to specify condition expressions in native BPMN flow objects. Therefore, we chose the Expression class to specify the process invariants.

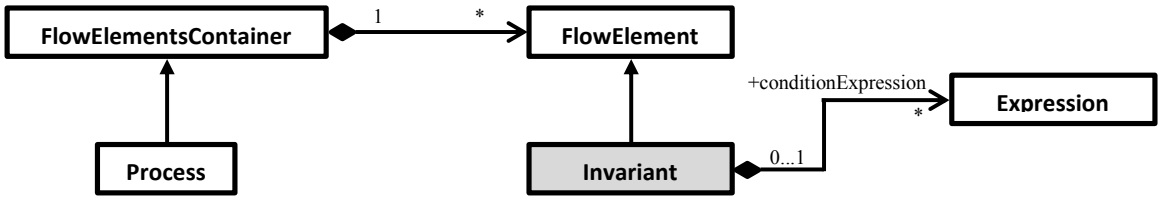

Fig. 2. Invariant class in extension of BPMN 2.0 specification.

Our approach requires an extension to the standard BPMN flow element. Therefore, we added the “Invariant” class through the XML Schema extension mechanism. This class represents the same characteristics as described in the BPMN flow element specification. Using the schema extension mechanism, we define a new complex type that inherits from tFlowElement. To maintain coherence, we define it as “tInvariant”.

Based on the domain model in figure 3, we now explain the requirements of the “tInvariant” complex type. The Invariant class has a collection of Expression elements. Its schema equivalent is the tExpression complex type. Additionally, the Expression class plays the conditionExpression role in association with our new “Invariant” class. Therefore, we add an element to the base extension and name it “conditionExpression” (of tExpression type), with an unlimited number of occurrences. Listing 1 displays the XML code that implements tInvariant and listing 2 displays condition expression syntax.

Listing 1. Invariant complex type definition.

```
<xsd:complexType name="tInvariant">
  <xsd:complexContent>
    <xsd:extension base="tFlowElement">
      <xsd:sequence>
        <xsd:element name="conditionExpression" type="tExpression" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Listing 2. Condition Expression syntax.

```
<conditionExpression><![CDATA[ObjectName Operator ValueToCompare]]></conditionExpression>
```

4.2. Context Object Construct

Process invariants define a set of conditions that are valid within the process. These invariants are meant to use data that should be available to the process. Namely, data gathered by IoT-related technologies. George *et al*. [18] define context as a state of environment conditions that are external to the process. The value of such conditions does not necessarily depend on the process workflow but the change of conditions may have an impact on the process execution.

Typically, context data is gathered during the process execution in specific points of the process workflow based on a synchronous request/response paradigm. However, this approach presents two issues according to our needs: a process invariant must be valid through all the process execution, and a process invariant must be aware of the environment changes as soon as they occur. Therefore, we also consider an asynchronous publish/subscribe paradigm to address them. Regarding these two paradigms we identify a set of requirements that are mandatory to satisfy our needs. Namely, a language construct to (1) specify how the process data is updated, i.e., specify the communication model, which can be request/response for synchronous scenarios or publish/subscribe for asynchronous scenarios and (2) represent environment data that is accessible within all the process workflow. Additionally, the request/response model requires (3) the address of the data provider and (4) the specification of the data property and (5) the timer to define the data update frequency. Moreover, the publish/subscribe model requires (6) the address of the data publisher and (7) the type of data that the object is interested in receiving (Temperature or Humidity, for instance).

Our goal is to represent context data within a BPMN process model. Therefore, we analysed the language BPMN constructs regarding the needs we identified. BPMN provides various constructs to handle items that model data within a process known as Item Aware Elements. They are similar to the variable construct in other modelling or programming languages. Namely, the BPMN specification contains the definition of Data Objects and Data Object References.

The Data Object is an item-aware element that must be contained within a Process or a Sub-Process element. It is the primary construct to model data within a Process flow. The lifecycle of a Data Object depends on the lifecycle of its parent process. Data Objects contained within a Process are initialised when a Process instance initiates and disposed when a Process instance ends. Additionally, a Data Object can only be access by its parent process or its sibling Flow Elements and their children. The Data Object Reference is an item-aware element that allows the representation of a Data Object in a process diagram. It is also a way to reuse Data Objects in the same diagram. Figure 4 illustrates the UML class representation of these elements according to BPMN specification with a white background.

Regarding the identified mandatory needs Data Objects satisfy the accessibility one. The other four needs demand a specific construct to specify them. Therefore, we propose Context Object, a new language construct that extends Data Object element with the attributes to address the remaining needs. Figure 4 illustrates the UML class representation of extension with a grey background.

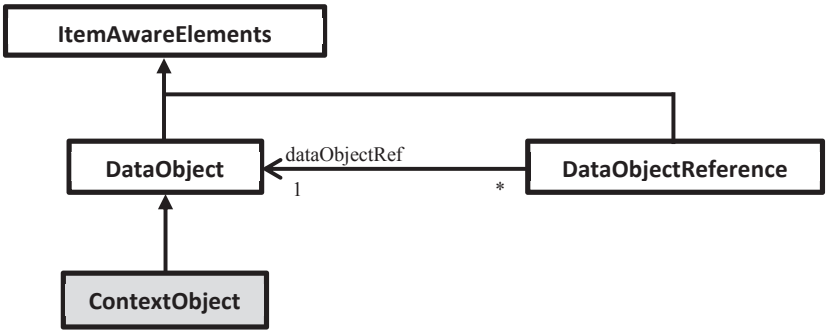

Fig. 3. Context Object class in extension of BPMN 2.0 specification.

Our approach requires an extension to the standard BPMN data object. Therefore, we added the "ContextObject" class through the XML Schema extension mechanism. This class represents the same characteristics as described in the BPMN flow element specification. Using the schema extension mechanism, we define a new complex type that inherits from tDataObject. To maintain coherence, we define it as "tContextObject".

We defined the requirements of the "tContexObject" type based on the needs that are not satisfied by Data Object regarding context data. We support two interaction mechanisms to update the object. One updates the object periodically using a synchronous request/response model. The other updates the object asynchronously with publish/subscribe interaction model. Therefore, we added the *CommunicationType* attribute: defines how the data object value will be updated, namely request/response or publish/subscribe.

Depending on which model, additional attributes must be declared. The synchronous model requires the definition of attributes to establish communication according to *WS-ResourceProperties* standard [21]. Namely: the endpoint address of the web service that provides context data (*sourceURL*); the resource property (*resourceProperty*) and the frequency of the update (refreshTimer). The asynchronous model requires the definition of attributes to establish a subscribe operation according to WS-Notifications standard. Namely: the endpoint address of the web service that provides the subscribe operation (*publisherURL*); and the subscription topic (*topic*). Listing 3 displays the syntax of contextObject for each of the communication models.

Listing 3. Context Object Syntax.

```
<dataObject name="objectName"
    comunicationType="publisher-subscriber"
    publisherURL="URL"
    topic="Topic"/>
<dataObject name="objectName"
    comunicationType="request-response"
    sourceURL="URL"
    resourceProperty="ResourceName"
    refreshTime="Time"/>
```

4.3. Example

With the new defined language constructs, we now explain how to use the complex types within the standard BPMN. Mainly, we focus on the condition expression which allows the definition of a process invariant and the necessary data objects that define how the data for the invariants will be obtained. Consider the need to define the following invariant conditions: temperature must maintain a value between 10 and 15, humidity must maintain a value higher than 50. Listing 4 represents a simplified process model sample with the two invariants defined in XML.

Listing 4. Invariant complex example in a process with a data data object.

```
<process id="Process_1" …>
  <dataObject id="DO_PROCESS_1_1" itemSubjectRef="xsd:int" name="temperatureObject"
      comunicationType="publisher-subscriber"
      publisher="http://192.168.1.52:8081/axis2/services/SensorService"
      topic="Temperature" />
  <dataObject id="DO_PROCESS_1_1" itemSubjectRef="xsd:int" name="humidityObject"
      comunicationType="request-response"
      sourceURL ="http://192.168.1.52:8081/axis2/services/SensorService"
      resourceProperty ="Humidity"
```

```
        refreshTime="60"/>
    <conditionExpression><![CDATA[humidityObject > 50]]></conditionExpression>
    <conditionExpression><![CDATA[temperatureObject > 10]]></conditionExpression>
    <conditionExpression><![CDATA[temperatureObject < 15]]></conditionExpression>
</process>
```

4.4. Graphical Notation

The specification of BPMN is a graphic oriented. Therefore, it must allow a representation of its components. So far, this paper proposed an extension that provides a way to the add invariants to the process. However, the extension elements do not necessarily have a graphical notation. Therefore, this section proposes a maker to inform that a process scope contains invariants. The design of the notation is inspired in Boolean operators used in expressions such as invariants.

A process invariant should be valid for a scope of the process model. Therefore, there is no restriction regarding the location of its graphical notation. Its freedom is similar to an annotation: it can be placed anywhere along the process. However, annotations can only be associated with a specific flow object, as process invariants should remain constant for the entire process. Therefore, their notation should be represented in a location that addresses the entire process. Figure 5 illustrates the invariants graphical notation designed in the overall poll of a process model. The design of the notation is inspired in Boolean operators used in expressions such as invariants.

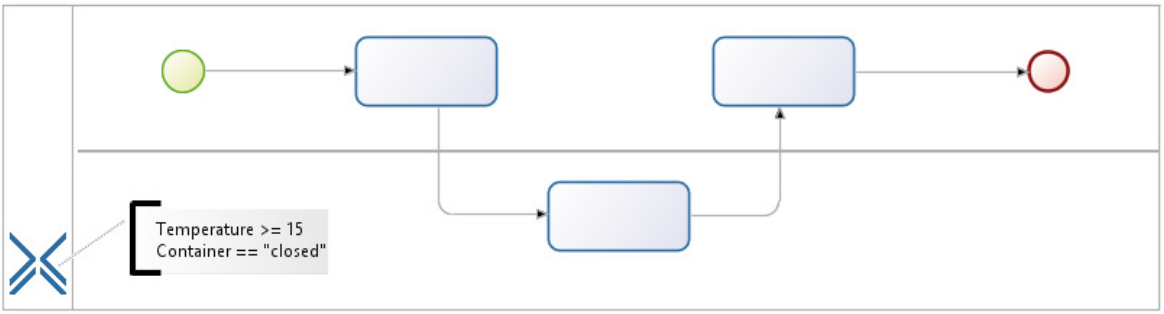

Fig. 4. Process model with Invariant graphical notation.

5. Conclusions and Future Work

The IoT faces many challenges from a business process modelling perspective, mainly because of its highly dynamic business processes like logistic functions. Typical solutions address this challenge by modelling exceptions and deviations. However, modelling exceptions is hard. Therefore, in this paper we presented “the right process” approach as an alternative to exception handling. We created the mechanism to define a set of conditions that guarantees the process is accordingly. Additionally, we named these conditions as invariants to the process.

BPMN provides coverage for more IoT specific properties than any other existing business process modelling standard. However, business processes require language specific constructs to define IoT conditions that native BPMN does not support. Therefore, we proposed an extension to model process invariants in BPMN. Namely, we created the Invariant element that extends from FlowElement element and contains a set of Expression elements. Additionally, we created the Context Object element that extends from Data Object and contains the behavioral information regarding how the data is retrieved from the environment, enhancing Data Objects with subscription models.

The motivation for this work is IoT aware business processes. However, the concept and modelling of invariants at process level can be useful in other contexts. As future work, we intend to merge the potential of process invariants with environment data obtained from IoT technologies concerning business process execution and monitoring. Namely, to use process invariants that monitor variables as they are updated during execution. The motivation for this approach follows the approach used by [5] for WS-BPEL.

Acknowledgements

This work was supported by FCT through funding of PATI project, ref. PTDC/EIAEIA/103751/2008, and LaSIGE Strategic Project, ref. PEst-OE/EEI/UI0408/2014.

References

[1] L. Baresi and S. Guinea. Towards dynamic monitoring of ws-bpel processes. In ICSOC 2005, Third International Conference of Service-Oriented Com- puting, volume 3826 of Lecture Notes in Computer Science, pages 269–282. Springer, 2005.
[2] A. Caracas and A. Bernauer. Compiling business process models for sen- sor networks. In Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on, pages 1 –8, june 2011.
[3] L. A. Clarke and D. S. Rosenblum. A historical perspective on runtime assertion checking in software development. SIGSOFT Softw. Eng. Notes, 31(3):25–37, May 2006.
[4] C. Decker, M. Berchtold, L. W. F. Chaves, M. Beigl, D. Roehr, T. Riedel, M. Beuster, T. Herzog, and D. Herzig. Cost-benefit model for smart items in the supply chain. In IOT'08: Proceedings of the 1st international conference on The internet of things, pages 155–172, Berlin, Heidelberg, 2008. Springer- Verlag.
[5] D. Domingos, R. Martinho, and C. Cândido. Flexibility in cross- organizational ws-bpel business processes. Procedia Technology, 9(0):584 – 595, 2013.
[6] S. Meyer, K. Sperner, C. Magerkurth, and J. Pasquier. Towards modeling real-world aware business processes. In Proceedings of the Second Interna- tional Workshop on Web of Things, WoT '11, pages 8:1–8:6, New York, NY, USA, 2011. ACM.
[7] OMG. Business process model and notation (bpmn) version 2.0. Technical Report dtc/2010-05-03, Object Management Group, 2010.
[8] OMG. Uml 2.3.1 ocl specification, available as omg document formal/2012- 01-01. Technical report, Object Management Group, Feb. 2012.
[9] G. Regev, I. Bider, and A. Wegmann. Defining business process flexibility with the help of invariants. Software Process: Improvement and Practice, 12(1):65–79, 2007.
[10] D. H. Akehurst: Validating BPEL Specifications using OCL, Technical Report No. 15-04, University of Kent, Canterbury, August 2004
[11] M. Meyer, A. Ruppen, C. Magerkurth. Internet of things-aware process modeling: integrating iot devices as business process resources. In Proceedings of the 25th international conference on Advanced Information Systems Engineering (CAiSE'13), Springer-Verlag, Berlin, Heidelberg, 84-98
[12] F. Casati. Towards business processes orchestrating the physical enterprise with wireless sensor networks. In Proceedings of the 2012 International Conference on Software Engineering (ICSE 2012). IEEE Press, Piscataway, NJ, USA, 1357-1360
[13] Bruno de Moura Araujo, Eber Assis Schmitz, Alexandre Luis Correa, and Antonio Juarez Alencar. 2010. A method for validating the compliance of business processes to business rules. In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10). ACM, New York, NY, USA, 145-149
[14] Tim van Eijndhoven, Maria-Eugenia Iacob, and María Laura Ponisio. 2008. Achieving Business Process Flexibility with Business Rules. In Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC '08). IEEE Computer Society, Washington, DC, USA, 95-104
[15] D. Knuplesch, M. Reichert, J. Mangler, S. Rinderle-Ma, and W. Fdhila. Towards Compliance of Cross-Organizational Processes and their Changes. In: Joint Workshop on Security in Business Processes (SBP 2012) in conjunction with BPM 2012 (2012)
[16] David Knuplesch, Linh Thao Ly, Stefanie Rinderle-Ma, Holger Pfeifer, and Peter Dadam. 2010. On enabling data-aware compliance checking of business process models. In Proceedings of the 29th international conference on Conceptual modeling (ER'10), Springer-Verlag, Berlin, Heidelberg, 332-346
[17] Ahmed Awad, Matthias Weidlich, and Mathias Weske. 2009. Specification, Verification and Explanation of Violation for Data Aware Compliance Rules. In Proceedings of the 7th International Joint Conference on Service-Oriented Computing (ICSOC-ServiceWave '09), Springer-Verlag, Berlin, Heidelberg, 500-515
[18] A. A. George and P. A. S. Ward. An architecture for providing con- text in ws-bpel processes. In Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds, CASCON '08, pages 22:289–22:303, New York, NY, USA, 2008. ACM
[19] S. Haller, S. Karnouskos and C. Schroth. The Internet of Things in an Enterprise Context. In Future Internet — FIS 2008: First Future Internet Symposium, FIS 2008 Vienna, Austria, September 29-30, 2008 Revised Selected Papers, pages 14–28, Berlin, Heidelberg, 2009. Springer-Verlag
[20] The Workflow Activity Model WAMO. In Proceedings of 3rd International Conference on Cooperative Information Systems, 87-98. Vienna, Austria, 1995.
[21] OASIS (2006b). Web services resource properties version 1.2. OASIS. URL: http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf.